Spring-MyBatis Hackathon

Points to Consider

- Does the Java code provide a complete and accurate solution for the stated requirements? Including, but not limited to the following:
  - Do the methods do what they are expected to do?
  - Is the right data returned in the right order?
  - Does the insert work as expected?
  - Are Spring and MyBatis used appropriately and correctly configured? Is auto-wiring used appropriately?
  - Was MyBatis used to develop the interaction with the database? Was the correct data retrieved from and added to the database?

- Do the JUnit tests provide adequate coverage of the functionality? Are exceptions used to manage errors that can occur in the code? For example:
  - Are there enough tests to fully cover the functionality?
  - Do the tests follow JUnit and TDD best practices?
  - Are there incorrect results that would pass the tests?
  - Are the tests self-contained and repeatable?
  - Does exception handling follow best practices?
  - Is exception handling tested as far as possible?

- Does the Java code follow best practices? For example:
  - Well laid out and easy to read
  - Short methods with descriptive names
  - DRY – Don’t Repeat Yourself
  - PIE

- Does the SQL meet the requirements? Does it follow best practices? For example: ◦ Are the correct results returned in the right order?
  - Do the DML statements insert, update, or delete the right rows?
  - Is code well laid out and readable?
  - If tables must be joined, are they joined on the right columns?

Scenario

You have been assigned to a development team that is querying and writing to a relational database. You have been provided with the following:

A) The HR database schema in your local Oracle instance and a script to recreate and populate if data corruption occurs. You may not change the tables in the schema.
B) A value class that represents the core data in the schema together. You may not change the existing fields and/or properties but follow the Product Owner's comments in the source file regarding adding properties. However, you may create additional methods in these classes.
C) An interface that represents a Data Access Object. You may not alter this interface in any way since another team has already started working on components that depend on the interface.
D) An interface that represents the MyBatis mapper. You may add methods to this interface to support any reasonable data operations, but do not modify the existing method. The existing must be bound to an appropriate SQL mapping.

Schema

- See the HR_SCHEMA_ERD.pdf in the diagram folder of your starter project.

Some points to consider:

- What do you need to test? Can you test both positive and negative outcomes?

Tasks

- Implement the DAO interface. Note the following additional requirements:
  - The getEmployeesAndJobs() method should retrieve a list of fully-populated Employees and their corresponding Jobs in employeeId order. All employees have an assigned job
  - To support hiring requirements to fill in-demand roles, the add method should assign the following jobs to new employees: Programmer, Accountant, Administrative Assistant, Human Resources Rep or Vice President

- Work TDD. You must provide tests for any functionality that you create. Your tests must be self-contained and repeatable.

- You will use MyBatis for all database interactions.

- The starter project includes an SQL files to re-create the appropriate tables if the schema objects get corrupted. You are welcome to use it.

- The pom for the starter project has been updated to include the libraries you will need, as far as your project team can determine. If you make changes to the pom, please comment them with an explanation.

- Use a Spring test context in the tests and ensure that the tests will all rollback data modifications.